Лабораторная работа №3.

Управляющие структуры.

Ишанова А.И. группа НФИ-02-19

Содержание

List of Figures

1 Цель работы

Освоить применение циклов функций и сторонних для Julia пакетов для решения задач линейной алгебры и работы с матрицами.

2 Задание

1. Используя Jupyter Lab, повторите примеры из раздела 3.2.
2. Выполните задания для самостоятельной работы (раздел 3.4). [1]

3 Выполнение лабораторной работы

3.1 Повторение примеров

1. Повторяем примеры с циклом while. (fig. 3.1 - fig. 3.2)

```
[1]: # пока n<10 прибавить к n единицу и распечатать значение:
n=0
while n < 10
n += 1
println(n)
end
```

```
1
2
3
4
5
6
7
8
9
10
```

Figure 3.1: Примеры с циклом while - 1

```
[2]: myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]

[2]: 5-element Vector{String}:
      "Ted"
      "Robyn"
      "Barney"
      "Lily"
      "Marshall"

[3]: i=1
     while i <= length(myfriends)
         friend = myfriends[i]
         println("Hi $friend, it's great to see you!")
         i += 1
     end

     Hi Ted, it's great to see you!
     Hi Robyn, it's great to see you!
     Hi Barney, it's great to see you!
     Hi Lily, it's great to see you!
     Hi Marshall, it's great to see you!
```

Figure 3.2: Примеры с циклом while - 2

2. Повторяем примеры с циклом for. (fig. 3.3 - fig. 3.4)

```
[4]: for n in 1:2:10
         println(n)
     end

     1
     3
     5
     7
     9
```

Figure 3.3: Примеры с циклом for - 1

```
[5]: myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
     for friend in myfriends
         println("Hi $friend, it's great to see you!")
     end

     Hi Ted, it's great to see you!
     Hi Robyn, it's great to see you!
     Hi Barney, it's great to see you!
     Hi Lily, it's great to see you!
     Hi Marshall, it's great to see you!
```

Figure 3.4: Примеры с циклом for - 2

3. Повторяем пример использования цикла for для создания двумерного массива, в котором значение каждой записи является суммой индексов строки и столбца. (fig. 3.5 - fig. 3.7)

```
[6]: # инициализация массива m x n из нулей:
     m, n = 5, 5
     A = fill(0, (m, n))
```

```
[6]: 5×5 Matrix{Int64}:
      0  0  0  0  0
      0  0  0  0  0
      0  0  0  0  0
      0  0  0  0  0
      0  0  0  0  0
```

```
[7]: # формирование массива, в котором значение каждой записи
     # является суммой индексов строки и столбца:
     for i in 1:m
         for j in 1:n
             A[i, j] = i + j
         end
     end
     A
```

```
[7]: 5×5 Matrix{Int64}:
      2  3  4  5   6
      3  4  5  6   7
      4  5  6  7   8
      5  6  7  8   9
      6  7  8  9  10
```

Figure 3.5: Пример цикла for для создания двумерного массива - 1ый вариант

```
[8]: # инициализация массива m x n из нулей:
     B = fill(0, (m, n))
     for i in 1:m, j in 1:n
         B[i, j] = i + j
     end
     B
```

```
[8]: 5×5 Matrix{Int64}:
      2  3  4  5   6
      3  4  5  6   7
      4  5  6  7   8
      5  6  7  8   9
      6  7  8  9  10
```

Figure 3.6: Пример цикла for для создания двумерного массива - 2ой вариант

```
[9]: C = [i + j for i in 1:m, j in 1:n]
     C
```

```
[9]: 5×5 Matrix{Int64}:
      2  3  4  5   6
      3  4  5  6   7
      4  5  6  7   8
      5  6  7  8   9
      6  7  8  9  10
```

Figure 3.7: Пример цикла for для создания двумерного массива - 3ий вариант

4. Повторяем пример с условными выражениями. (fig. 3.8)

[11]:
```
# используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
N = 15
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
FizzBuzz
```

[12]:
```
N = 5
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
Buzz
```

[13]:
```
N = 3
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
Fizz
```

[14]:
```
N = 1
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
1
```

Figure 3.8: Пример с условными выражениями

5. Повторяем пример с тернарным оператором. (fig. 3.9)

```
[15]: # Пример использования тернарного оператора:
      x=5
      y = 10
      (x > y) ? x : y

[15]: 10
```

Figure 3.9: Пример с тернарным оператором

6. Повторяем примеры задания функции. (fig. 3.10)

```
[16]: function sayhi(name)
          println("Hi $name, it's great to see you!")
      end

      # функция возведения в квадрат:
      function f(x)
          x^2
      end

[16]: f (generic function with 1 method)

[19]: sayhi("C-3PO")

      Hi C-3PO, it's great to see you!

[20]: f(42)

[20]: 1764

[21]: sayhi2(name) = println("Hi $name, it's great to see you!")
      f2(x) = x^2

[21]: f2 (generic function with 1 method)

[22]: sayhi2("C-3PO")

      Hi C-3PO, it's great to see you!

[23]: f2(42)

[23]: 1764

[24]: sayhi3 = name -> println("Hi $name, it's great to see you!")
      f3 = x -> x^2

[24]: #5 (generic function with 1 method)

[25]: sayhi3("C-3PO")

      Hi C-3PO, it's great to see you!

[26]: f3(42)

[26]: 1764
```

Figure 3.10: Примеры с заданием функции

7. Повторяем примеры функций с восклицательным знаком. (fig. 3.11)

```
[27]: v = [3, 5, 2]
      sort(v)
      v

[27]: 3-element Vector{Int64}:
       3
       5
       2

[28]: v = [3, 5, 2]
      sort!(v)
      v

[28]: 3-element Vector{Int64}:
       2
       3
       5
```

Figure 3.11: Пример с функциями sort и sort!

8. Повторяем примеры с map(). (fig. 3.12)

```
[29]: f(x) = x^2
      map(f, [1, 2, 3])

[29]: 3-element Vector{Int64}:
       1
       4
       9

[30]: x -> x^3
      map(x -> x^3, [1, 2, 3])

[30]: 3-element Vector{Int64}:
        1
        8
       27
```

Figure 3.12: Примеры с map()

9. Повторяем примеры с broadcast(). (fig. 3.13 - fig. 3.15)

```
[31]: f(x) = x^2
      broadcast(f, [1, 2, 3])

[31]: 3-element Vector{Int64}:
       1
       4
       9

[32]: f.([1, 2, 3])

[32]: 3-element Vector{Int64}:
       1
       4
       9
```

Figure 3.13: Пример с broadcast() с векторами

```
[33]: # Задаём матрицу A:
      A = [i + 3*j for j in 0:2, i in 1:3]

[33]: 3×3 Matrix{Int64}:
       1  2  3
       4  5  6
       7  8  9

[34]: # Вызываем функцию f возведения в квадрат
      f(A)

[34]: 3×3 Matrix{Int64}:
        30   36   42
        66   81   96
       102  126  150

[35]: B = f.(A)

[35]: 3×3 Matrix{Int64}:
         1   4   9
        16  25  36
        49  64  81
```

Figure 3.14: Пример с broadcast() с матрицами

```
[36]: A .+ 2 .* f.(A) ./ A

[36]: 3×3 Matrix{Float64}:
        3.0   6.0   9.0
       12.0  15.0  18.0
       21.0  24.0  27.0

[37]: @. A + 2 * f(A) / A

[37]: 3×3 Matrix{Float64}:
        3.0   6.0   9.0
       12.0  15.0  18.0
       21.0  24.0  27.0

[38]: broadcast(x -> x + 2 * f(x) / x, A)

[38]: 3×3 Matrix{Float64}:
        3.0   6.0   9.0
       12.0  15.0  18.0
       21.0  24.0  27.0
```

Figure 3.15: Пример с broadcast() с точечным синтаксисом

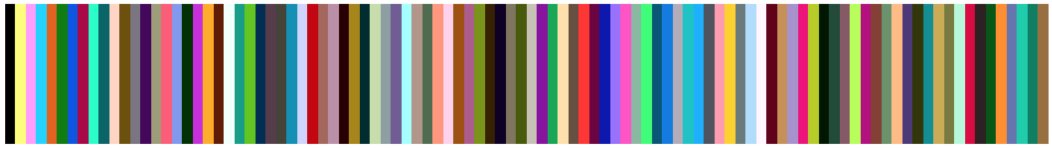
10. Установка пакета Colors. (fig. 3.16)

```
julia> using Pkg

julia> Pkg.add("Colors")
    Updating registry at `~/.julia/registries/General.toml`
   Resolving package versions...
    Updating `~/.julia/environments/v1.8/Project.toml`
  [5ae59095] + Colors v0.12.8
    Updating `~/.julia/environments/v1.8/Manifest.toml`
  [3da002f7] + ColorTypes v0.11.4
  [5ae59095] + Colors v0.12.8
  [53c48c17] + FixedPointNumbers v0.8.4
  [189a3867] + Reexport v1.2.2
  [37e2e46d] + LinearAlgebra
  [2f01184e] + SparseArrays
  [10745b16] + Statistics
  [e66e0078] + CompilerSupportLibraries_jll v0.5.2+0
  [4536629a] + OpenBLAS_jll v0.3.20+0
  [8e850b90] + libblastrampoline_jll v5.1.1+0
Precompiling project...
  7 dependencies successfully precompiled in 9 seconds. 19 already precompiled.

julia>
```

Figure 3.16: Установка пакета Colors

11. Повторяем пример с палитрой и рандомной матрицей сэлементами-цветами. (fig. 3.17)

```
[39]: using Colors
      palette = distinguishable_colors(100)
```



```
[40]: rand(palette, 3, 3)
```

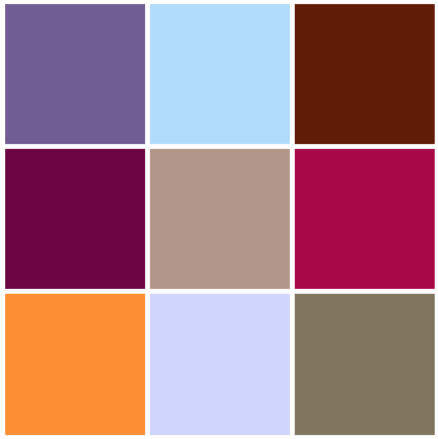


Figure 3.17: Примеры с пакетом Colors

3.2 Задания для самостоятельной работы

1. Используя циклы while и for: – вывели на экран целые числа от 1 до 100 и напечатали их квадраты (fig. 3.18-fig. 3.19);

[42]:
```
# 1.

n = 1
while n<=100
    println("$n ", n^2)
    n = n+1
end
```

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
14 196
15 225
16 256
17 289
18 324
19 361
20 400
21 441
```

Figure 3.18: Вывод чисел и их квадратов через цикл while

```
: for n in 1:100
      println(n, " ", n^2)
  end
```

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
```

Figure 3.19: Вывод чисел и их квадратов через цикл for

– создали словарь squares,который содержит целые числа в качестве ключей и квадраты в качестве их пар-значений (fig. 3.20-fig. 3.21);

[44]:
```
squares = Dict()
for i in 1:100
    squares[i] = i^2
end
pairs(squares)
```

[44]:
```
Dict{Any, Any} with 100 entries:
  5  => 25
  56 => 3136
  35 => 1225
  55 => 3025
  60 => 3600
  30 => 900
  32 => 1024
  6  => 36
  67 => 4489
  45 => 2025
  73 => 5329
  64 => 4096
  90 => 8100
  4  => 16
  13 => 169
  54 => 2916
  63 => 3969
  86 => 7396
  91 => 8281
  62 => 3844
  58 => 3364
  52 => 2704
  12 => 144
  28 => 784
  75 => 5625
  ⋮  => ⋮
```

Figure 3.20: Создание словаря squares через цикл for

```
[48]: n = 1
      while n<=100
          squares[n] = n^2
          n = n+1
      end
      pairs(squares)
```

```
[48]: Dict{Any, Any} with 100 entries:
        5  => 25
        56 => 3136
        35 => 1225
        55 => 3025
        60 => 3600
        30 => 900
        32 => 1024
        6  => 36
        67 => 4489
        45 => 2025
        73 => 5329
        64 => 4096
        90 => 8100
        4  => 16
        13 => 169
        54 => 2916
        63 => 3969
        86 => 7396
        91 => 8281
        62 => 3844
        58 => 3364
        52 => 2704
        12 => 144
        28 => 784
        75 => 5625
        ⋮  => ⋮
```

Figure 3.21: оздание словаря squares через цикл while

– создали массив squares_arr, содержащий квадраты всех чисел от 1 до 100 (fig. 3.22-fig. 3.23).

```
[49]: squares_arr = []
      for i in 1:100
          append!(squares_arr, i^2)
      end
      squares_arr
```

```
[49]: 100-element Vector{Any}:
        1
        4
        9
       16
       25
       36
       49
       64
       81
      100
      121
      144
      169
        ⋮
     7921
     8100
     8281
     8464
     8649
     8836
     9025
     9216
     9409
     9604
     9801
    10000
```

Figure 3.22: Создание массива squares_arr через цикл for

```
[50]: squares_arr = []
      n = 1
      while n <= 100
          append!(squares_arr, n^2)
          n = n+1
      end
      squares_arr
```

```
[50]: 100-element Vector{Any}:
        1
        4
        9
       16
       25
       36
       49
       64
       81
      100
      121
      144
      169
        ⋮
     7921
     8100
     8281
     8464
     8649
     8836
     9025
     9216
     9409
     9604
     9801
    10000
```

Figure 3.23: Создание массива squares_arr через цикл while

2. Написали условный оператор, который печатает число, если число чётное, и строку «нечётное», если число нечётное. Переписали код, используя тернарный оператор. (fig. 3.24)

```
[278]: # 2

       N = 2
       if (N%2==0)
           println(N)
       else
           println("нечётное")
       end
```

```
2
```

```
[279]: (N%2==0) ? println(N) : println("нечётное")
```

```
2
```

Figure 3.24: Написание условного оператора

3. Написали функцию add_one, которая добавляет 1 к своему входу. (fig. 3.25)

```
[54]: # 3

function add_one(x)
    x+1
end

add_one(1)
```

```
[54]: 2
```

Figure 3.25: Написание функции add_one

4. Использовали broadcast() для задания матрицы A, каждый элемент которой увеличивается на единицу по сравнению с предыдущим. (fig. 3.26)

```
[82]: # 4

A = ones(5,5)
```

```
[82]: 5×5 Matrix{Float64}:
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
```

```
[83]: for i in 1:5, j in 1:5
    A[i, j:5] = broadcast(add_one, A[i, j:5])
    if (i!=5 && j==5)
        A[i+1, :] = fill(A[i,5], (1,5))
    end
end
A
```

```
[83]: 5×5 Matrix{Float64}:
  2.0   3.0   4.0   5.0   6.0
  7.0   8.0   9.0  10.0  11.0
 12.0  13.0  14.0  15.0  16.0
 17.0  18.0  19.0  20.0  21.0
 22.0  23.0  24.0  25.0  26.0
```

Figure 3.26: Написание условного оператора

5. Задали матрицу

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}$$

- нашли A^3 (fig. 3.27)
- заменили третий столбец на сумму второго и третьего столбцов (fig. 3.27)

```
[90]: # 5

      A = [ [1, 5, -2] [1, 2, -1] [3, 6, -3]]

[90]: 3×3 Matrix{Int64}:
        1   1   3
        5   2   6
       -2  -1  -3

[92]: A^3

[92]: 3×3 Matrix{Int64}:
       0  0  0
       0  0  0
       0  0  0

[94]: A[:, 3] = A[:,2]+A[:,3]
      A

[94]: 3×3 Matrix{Int64}:
        1   1   4
        5   2   8
       -2  -1  -4
```

Figure 3.27: Задание матрицы A и операции с ней

6. Создали матрицу B с элементами $B_{i1} = 10, B_{i2} = -10, B_{i3} = 10, i = 1, 2, 3, ..., 15$. Нашли матрицу $C = B^T B$. (fig. 3.28)

```
[103]: # 6

       B = repeat([10 -10 10], 15)
```

```
[103]: 15×3 Matrix{Int64}:
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
        10  -10  10
```

```
[104]: C = B' * B
```

```
[104]: 3×3 Matrix{Int64}:
         1500  -1500   1500
        -1500   1500  -1500
         1500  -1500   1500
```

Figure 3.28: Задание матрицы B и расчет матрицы C

7. Создали матрицу Z размерности 6×6,все элементы которой равны нулю, и матрицу E, все элементы которой равны 1. Используя цикл while или for и закономерности расположения элементов, создали следующие матрицы размерности 6 × 6:

    - Z1 (fig. 3.29)

[115]:
```
# 7

Z = zeros(6,6)
E = ones(6,6)

Z1 = zeros(6,6)

for i in 1:6, j in 1:6
    if (abs(i-j)==1)
        Z1[i,j] = E[i,j]
    end
end
Z1
```

[115]:
```
6×6 Matrix{Float64}:
 0.0  1.0  0.0  0.0  0.0  0.0
 1.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  1.0  0.0  0.0
 0.0  0.0  1.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  1.0
 0.0  0.0  0.0  0.0  1.0  0.0
```

Figure 3.29: Задание матрицы Z1

- Z2 (@fig:030)

[118]:
```
Z2 = zeros(6,6)
for i in 1:6, j in 1:6
    if (abs(i-j)==2 || i==j)
        Z2[i,j] = E[i,j]
    end
end
Z2
```

[118]:
```
6×6 Matrix{Float64}:
 1.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  1.0  0.0  0.0
 1.0  0.0  1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0  0.0  1.0
 0.0  0.0  1.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  1.0
```

Figure 3.30: Задание матрицы Z2

- Z3 (@fig:031)

```
[128]: Z3 = zeros(6,6)
       for i in 1:6, j in 1:6
           if (j == 7-i || j == 5-i || j == 9-i)
               Z3[i,j] = E[i,j]
           end
       end
       Z3
```

```
[128]: 6×6 Matrix{Float64}:
        0.0  0.0  0.0  1.0  0.0  1.0
        0.0  0.0  1.0  0.0  1.0  0.0
        0.0  1.0  0.0  1.0  0.0  1.0
        1.0  0.0  1.0  0.0  1.0  0.0
        0.0  1.0  0.0  1.0  0.0  0.0
        1.0  0.0  1.0  0.0  0.0  0.0
```

Figure 3.31: Задание матрицы Z3

- Z4 (@fig:032)

```
[133]: Z4 = zeros(6,6)
       for i in 1:6, j in 1:6
           if (j == i || abs(i-j)==2 || abs(i-j)==4 )
               Z4[i,j] = E[i,j]
           end
       end
       Z4
```

```
[133]: 6×6 Matrix{Float64}:
        1.0  0.0  1.0  0.0  1.0  0.0
        0.0  1.0  0.0  1.0  0.0  1.0
        1.0  0.0  1.0  0.0  1.0  0.0
        0.0  1.0  0.0  1.0  0.0  1.0
        1.0  0.0  1.0  0.0  1.0  0.0
        0.0  1.0  0.0  1.0  0.0  1.0
```

Figure 3.32: Задание матрицы Z4

8. Написали свою функцию эквивалентную функции outer() в языке R. (fig. 3.33)

```
[218]: #8
       function outer(x,y,operation)
           res = zeros(size(x)[1], size(y)[2])
           for i in 1:size(x)[1], j in 1:size(y)[2], k in 1:size(x)[2]
               res[i,j]+=operation(x[i,k],y[k,j])
           end
           return res
       end
```

```
[218]: outer (generic function with 1 method)
```

Figure 3.33: Задание функции outer()

Используя нашу функцию outer(), задали следующие матрицы:

- A1 (@fig:034)

```
[228]: aa = collect(0:4)
       aa = reshape(aa, (size(aa,1), size(aa,2)))
```

```
[228]: 5×1 Matrix{Int64}:
        0
        1
        2
        3
        4
```

```
[229]: A1 = outer(aa, aa', +)
```

```
[229]: 5×5 Matrix{Float64}:
        0.0  1.0  2.0  3.0  4.0
        1.0  2.0  3.0  4.0  5.0
        2.0  3.0  4.0  5.0  6.0
        3.0  4.0  5.0  6.0  7.0
        4.0  5.0  6.0  7.0  8.0
```

Figure 3.34: Задание матрицы A1

- A2 (fig. 3.35)

```
[231]: A2 = outer(aa, collect(1:5)', ^)
```

```
[231]: 5×5 Matrix{Float64}:
        0.0   0.0   0.0    0.0     0.0
        1.0   1.0   1.0    1.0     1.0
        2.0   4.0   8.0   16.0    32.0
        3.0   9.0  27.0   81.0   243.0
        4.0  16.0  64.0  256.0  1024.0
```

Figure 3.35: Задание матрицы A2

- A3 (fig. 3.36)

```
[239]: A3 = .%(outer(aa, aa',+),5)
```

```
[239]: 5×5 Matrix{Float64}:
        0.0  1.0  2.0  3.0  4.0
        1.0  2.0  3.0  4.0  0.0
        2.0  3.0  4.0  0.0  1.0
        3.0  4.0  0.0  1.0  2.0
        4.0  0.0  1.0  2.0  3.0
```

Figure 3.36: Задание матрицы A3

- A4 (fig. 3.37)

```
[241]: bb = collect(0:9)
       bb = reshape(bb, (size(bb,1), size(bb,2)))
       A4 = .%(outer(bb, bb',+),10)
```

```
[241]: 10×10 Matrix{Float64}:
        0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
        1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0
        2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0
        3.0  4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0
        4.0  5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0
        5.0  6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0
        6.0  7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0
        7.0  8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0
        8.0  9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0
        9.0  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0
```

Figure 3.37: Задание матрицы A4

- A5 (fig. 3.38)

```
[246]: cc = collect(0:8)
       dd = collect(9:-1:1)
       cc = reshape(cc, (size(cc,1), size(cc,2)))
       dd = reshape(dd, (size(dd,1), size(dd,2)))

       A5 = .%(outer(cc, dd', +),9)
```

```
[246]: 9×9 Matrix{Float64}:
        0.0  8.0  7.0  6.0  5.0  4.0  3.0  2.0  1.0
        1.0  0.0  8.0  7.0  6.0  5.0  4.0  3.0  2.0
        2.0  1.0  0.0  8.0  7.0  6.0  5.0  4.0  3.0
        3.0  2.0  1.0  0.0  8.0  7.0  6.0  5.0  4.0
        4.0  3.0  2.0  1.0  0.0  8.0  7.0  6.0  5.0
        5.0  4.0  3.0  2.0  1.0  0.0  8.0  7.0  6.0
        6.0  5.0  4.0  3.0  2.0  1.0  0.0  8.0  7.0
        7.0  6.0  5.0  4.0  3.0  2.0  1.0  0.0  8.0
        8.0  7.0  6.0  5.0  4.0  3.0  2.0  1.0  0.0
```

Figure 3.38: Задание матрицы A5

9. Решили систему уравнений (fig. 3.39)

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = 7 \\ 2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 = -1 \\ 3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 = -3 \\ 4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 = 5 \\ 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 17 \end{cases}$$

```
[249]: #9

       A = [1 2 3 4 5; 2 1 2 3 4; 3 2 1 2 3; 4 3 2 1 2; 5 4 3 2 1]

[249]: 5×5 Matrix{Int64}:
        1  2  3  4  5
        2  1  2  3  4
        3  2  1  2  3
        4  3  2  1  2
        5  4  3  2  1

[250]: y = [ 7 -1 -3 5 17]

[250]: 1×5 Matrix{Int64}:
        7  -1  -3  5  17

[251]: function solveSLAU(A,y)
           inv(A)*y
       end

[251]: solveSLAU (generic function with 1 method)

[256]: X = solveSLAU(A,y')

[256]: 5×1 Matrix{Float64}:
        -1.9999999999999996
         3.0
         5.0
         2.0
        -4.0
```

Figure 3.39: Решение системы уравнений

10. Создали матрицу M размерности 6×10, элементами которой являются целые числа, выбранные случайным образом с повторениями из совокупности 1, 2, … , 10. (fig. 3.40)

- нашли число элементов в каждой строке матрицы, которые больше числа N=4 (fig. 3.40)

```
[257]: # 10

       M = rand(1:10, 6, 10)
```

```
[257]: 6×10 Matrix{Int64}:
        6  7  1   8  8  5   6   8   6  4
        4  1  1  10  4  2   7   2   4  1
        2  5  9  10  2  8   5   1   7  1
        3  3  7   3  9  8   8   2   4  3
        5  9  4   9  6  7  10   2   5  3
        4  4  5   9  9  8   2  10  10  5
```

```
[263]: N = 4
       for i in 1:size(M,1)
           k = 0
       for j in 1:size(M,2)
           if (M[i,j] > N)
                   k = k+1
           end
           end
           println(i, " ", k )
       end
```

```
1 8
2 2
3 6
4 4
5 7
6 7
```

Figure 3.40: Задание матрицы M и подсчет элементов больших 4 построчно

- определили в каких строках число M=7 встречается ровно 2 раза (таких нет, поэтому ничего не выводится) (fig. 3.41)

```
[277]: NM = 7
       for i in 1:size(M,1)
           if (length(findall(M[i,:] .== NM))==2)
               println(i)
           end
       end
```

```
[271]: K = 75
```

Figure 3.41: Определние в каких строках число M=7 встречается ровно 2 раза

- определили все пары столбцов матрицы, сумма элементов которых больше K=75 (fig. 3.42)

[271]:
```
K = 75
for i in 1:size(M,1)
    for u in i+1:size(M,1)
        if (sum(M[i,:]+M[u,:]) > K)
            println(i, " ", u)
        end
    end
end
```

```
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 6
```

Figure 3.42: Определние всех пар столбцов матрицы, сумма элементов которых больше K=75

11. Вычислили (fig. 3.43)

$$\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+j)}$$

и

$$\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+ij)}$$

[272]:
```
# 11

r1 = 0
for i in 1:20, j in 1:5
    r1 += i^4/(3+j)
end
r1
```

[272]:
```
639215.2833333334
```

[273]:
```
r2 = 0
for i in 1:20, j in 1:5
    r2 += i^4/(3+i*j)
end
r2
```

[273]:
```
89912.02146097136
```

Figure 3.43: вычисление выражений заданных в задании 11

4 Листинг

```
# -*- coding: utf-8 -*-
# ---
# jupyter:
#   jupytext:
#     text_representation:
#       extension: .jl
#       format_name: light
#       format_version: '1.5'
#       jupytext_version: 1.14.1
#   kernelspec:
#     display_name: Julia 1.8.2
#     language: julia
#     name: julia-1.8
# ---

# пока n<10 прибавить к n единицу и распечатать значение:
n=0
while n < 10
n += 1
println(n)
end
```

```
myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]

i=1
while i <= length(myfriends)
    friend = myfriends[i]
    println("Hi $friend, it's great to see you!")
    i += 1
end

for n in 1:2:10
    println(n)
end

myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]
for friend in myfriends
    println("Hi $friend, it's great to see you!")
end

# инициализация массива m x n из нулей:
m, n = 5, 5
A = fill(0, (m, n))

# формирование массива, в котором значение каждой записи
# является суммой индексов строки и столбца:
for i in 1:m
    for j in 1:n
        A[i, j] = i + j
    end
end
```

```
A

# инициализация массива m x n из нулей:
B = fill(0, (m, n))
for i in 1:m, j in 1:n
    B[i, j] = i + j
end
B

C = [i + j for i in 1:m, j in 1:n]
C

# используем `&&` для реализации операции "AND"
# операция % вычисляет остаток от деления
N = 15
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end

N = 5
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
```

```
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end

N = 3
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end

N = 1
if (N % 3 == 0) && (N % 5 == 0)
    println("FizzBuzz")
elseif N % 3 == 0
    println("Fizz")
elseif N % 5 == 0
    println("Buzz")
else
    println(N)
end
```

```
# Пример использования тернарного оператора:
x=5
y = 10
(x > y) ? x : y

# +
function sayhi(name)
    println("Hi $name, it's great to see you!")
end

# функция возведения в квадрат:
function f(x)
    x^2
end
# -

sayhi("C-3PO")

f(42)

sayhi2(name) = println("Hi $name, it's great to see you!")
f2(x) = x^2

sayhi2("C-3PO")

f2(42)

sayhi3 = name -> println("Hi $name, it's great to see you!")
f3 = x -> x^2
```

```
sayhi3("C-3PO")

f3(42)

v = [3, 5, 2]
sort(v)
v

v = [3, 5, 2]
sort!(v)
v

f(x) = x^2
map(f, [1, 2, 3])

x -> x^3
map(x -> x^3, [1, 2, 3])

f(x) = x^2
broadcast(f, [1, 2, 3])

f.([1, 2, 3])

# Задаём матрицу A:
A = [i + 3*j for j in 0:2, i in 1:3]

# Вызываем функцию f возведения в квадрат
f(A)
```

```
B = f.(A)

A .+ 2 .* f.(A) ./ A

@. A + 2 * f(A) / A

broadcast(x -> x + 2 * f(x) / x, A)

using Colors
palette = distinguishable_colors(100)

# # rand(palette, 3, 3)

# +
# ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

# +
# 1.

n = 1
while n<=100
    println("$n ", n^2)
    n = n+1
end

# -

# for n in 1:100
```

```
#     println("$n ", n^2)
# end

# for n in 1:100
#     println(n, " ", n^2)
# end

squares = Dict()
for i in 1:100
    squares[i] = i^2
end
pairs(squares)

n = 1
while n<=100
    squares[n] = n^2
    n = n+1
end
pairs(squares)

squares_arr = []
for i in 1:100
    append!(squares_arr, i^2)
end
squares_arr

# +
# 2
```

```
N = 2
if (N%2==0)
    println(N)
else
    println("нечётное")
end
# -

(N%2==0) ? println(N) : println("нечётное")

# +
# 3

function add_one(x)
    x+1
end

add_one(1)

# +
# 4

A = ones(5,5)
# -

for i in 1:5, j in 1:5
    A[i, j:5] = broadcast(add_one, A[i, j:5])
    if (i!=5 && j==5)
        A[i+1, :] = fill(A[i,5], (1,5))
```

```
    end
end
A

# +
# 5

A = [ [1, 5, -2] [1, 2, -1] [3, 6, -3]]
# -

A^3

A[:, 3] = A[:,2]+A[:,3]
A

# +
# 6

B = repeat([10 -10 10], 15)
# -

C = B' * B

# +
# 7

Z = zeros(6,6)
E = ones(6,6)
```

```
Z1 = zeros(6,6)

for i in 1:6, j in 1:6
    if (abs(i-j)==1)
        Z1[i,j] = E[i,j]
    end
end
Z1

# + tags=[]
Z2 = zeros(6,6)
for i in 1:6, j in 1:6
    if (abs(i-j)==2 || i==j)
        Z2[i,j] = E[i,j]
    end
end
Z2
# -

Z3 = zeros(6,6)
for i in 1:6, j in 1:6
    if (j == 7-i || j == 5-i || j == 9-i)
        Z3[i,j] = E[i,j]
    end
end
Z3

Z4 = zeros(6,6)
for i in 1:6, j in 1:6
```

```
    if (j == i || abs(i-j)==2 || abs(i-j)==4 )
        Z4[i,j] = E[i,j]
    end
end
Z4

#8
function outer(x,y,operation)
    res = zeros(size(x)[1], size(y)[2])
    for i in 1:size(x)[1], j in 1:size(y)[2], k in 1:size(x)[2]
        res[i,j]+=operation(x[i,k],y[k,j])
    end
    return res
end

aa = collect(0:4)
aa = reshape(aa, (size(aa,1), size(aa,2)))

A1 = outer(aa, aa', +)

A2 = outer(aa, collect(1:5)', ^)

A3 = .%(outer(aa, aa',+),5)

bb = collect(0:9)
bb = reshape(bb, (size(bb,1), size(bb,2)))
A4 = .%(outer(bb, bb',+),10)

# +
```

```
cc = collect(0:8)
dd = collect(9:-1:1)
cc = reshape(cc, (size(cc,1), size(cc,2)))
dd = reshape(dd, (size(dd,1), size(dd,2)))

A5 = .%(outer(cc, dd', +),9)

# +
#9

A = [1 2 3 4 5; 2 1 2 3 4; 3 2 1 2 3; 4 3 2 1 2; 5 4 3 2 1]
# -

y = [ 7 -1 -3 5 17]

function solveSLAU(A,y)
    inv(A)*y
end

X = solveSLAU(A,y')

# +
# 10

M = rand(1:10, 6, 10)
# -

N = 4
for i in 1:size(M,1)
```

```
    k = 0
for j in 1:size(M,2)
    if (M[i,j] > N)
            k = k+1
    end
    end
    println(i, " ", k )
end

NM = 7
for i in 1:size(M,1)
    if (length(findall(M[i,:] .== NM))==2)
        println(i)
    end
end

K = 75
for i in 1:size(M,1)
    for u in i+1:size(M,1)
        if (sum(M[i,:]+M[u,:]) > K)
            println(i, " ", u)
        end
    end
end

# +
# 11

r1 = 0
```

```
for i in 1:20, j in 1:5
    r1 += i^4/(3+j)
end
r1
# -

r2 = 0
for i in 1:20, j in 1:5
    r2 += i^4/(3+i*j)
end
r2
```

5 Вывод

В ходе выполнения лабораторной работы на примерах были изучены управляющие структуры, мы ознакомились с циклами for и while, условными операторами, тернарными операторами, функциями map() и broadcast(), способами написания своих функций и пакетом Colors на языке Julia. С помощью полученных знаний были решены задачи для самостоятельной работы.

6 Библиография

1. Методические материалы курса.